Neural Architecture Search with Random Labels

Xuanyang Zhang Pengfei Hou Xiangyu Zhang* Jian Sun
MEGVII Technology
{zhangxuanyang,houpengfei,zhangxiangyu,sunjian}@megvii.com

Abstract

In this paper, we investigate a new variant of neural architecture search (NAS) paradigm – searching with random labels (RLNAS). The task sounds counter-intuitive for most existing NAS algorithms since random label provides few information on the performance of each candidate architecture. Instead, we propose a novel NAS framework based on ease-of-convergence hypothesis, which requires only random labels during searching. The algorithm involves two steps: first, we train a SuperNet using random labels; second, from the SuperNet we extract the subnetwork whose weights change most significantly during the training. Extensive experiments are evaluated on multiple datasets (e.g. NAS-Bench-201 and ImageNet) and multiple search spaces (e.g. DARTS-like and MobileNet-like). Very surprisingly, RLNAS achieves comparable or even better results compared with state-of-the-art NAS methods such as PC-DARTS, Single Path One-Shot, even though the counterparts utilize full ground truth labels for searching. We hope our finding could inspire new understandings on the essential of NAS.

1. Introduction

Recent years *Neural Architecture Search* [49, 2, 50, 47, 48, 29, 35, 38, 10] (NAS) has received much attention in the community as its superior performances over human-designed architectures on a variety of tasks such as image classification [38, 39, 19], object detection [10, 16] and semantic segmentation [27]. In general, most existing NAS frameworks can be summarized as a *nested bilevel optimization*, formulated as follows:

$$a^\star = \underset{a\in\mathcal{A}}{\operatorname{argmax}}\, \text{Score}\left(a, \mathbf{W}_a^\star\right) \tag{1}$$

$$\text{s.t.} \quad \mathbf{W}_a^\star = \underset{\mathbf{W}}{\operatorname{argmin}}\, \mathcal{L}\left(a, \mathbf{W}\right), \tag{2}$$

where a is a candidate architecture with weights $\mathbf{W}_a$ sampled from the search space $\mathcal{A}$; $\mathcal{L}(\cdot)$ represents the *training loss*; $\text{Score}(\cdot)$ means the performance indicator (e.g. accuracy in supervised NAS algorithms or *pretext task* scores in unsupervised NAS frameworks [28]) evaluated on the *validation set*. Briefly speaking, the NAS paradigm aims to search for the architecture which obtains the best validation performance, thus we name it *performance-based NAS* in the remaining text.

Despite the great success, to understand why and how *performance-based NAS* works is still an open question. Especially, the mechanism how NAS algorithms discover good architectures from the huge search space is well worth study. A recent literature [37] analyzes the searching results under cell-based search spaces and reveals that existing performance-based methods tend to favor architectures with fast convergence. Although Shu *et al.* [37] further empirically find that architectures with fast convergence can not achieve the highest generalization performance, the fast convergence connection pattern still implies that there may exist high correlations between architectures with fast convergence and the ones with high performance (named *ease-of-convergence hypothesis* for short). Inspired by the hypothesis, we propose an alternative NAS paradigm, *convergence-based NAS*, as follows:

$$a^\star = \underset{a\in\mathcal{A}}{\operatorname{argmax}}\, \text{Convergence}\left(a, \mathbf{W}_a^\star\right) \tag{3}$$

$$\text{s.t.} \quad \mathbf{W}_a^\star = \underset{\mathbf{W}}{\operatorname{argmin}}\, \mathcal{L}\left(a, \mathbf{W}\right), \tag{4}$$

where $\text{Convergence}(\cdot)$ is a certain indicator to measure the speed of convergence; other notations follow the same definitions as in Eq. 1, 2.

In this paper we mainly investigate *convergence-based NAS* frameworks, which is rarely *explicitly* explored in previous works to our knowledge. First of all, we study the role of *labels* in both frameworks. In *performance-based NAS*, we notice that *feasible labels* are critical in both search steps: for Eq. 1 step, since we need to select the architecture with the highest validation performance, reasonable labels such as ground truths or at least carefully-designed *pretext task* (e.g. rotation prediction [17]) labels in unsupervised

*Corresponding author. This work is supported by The National Key Research and Development Program of China (No.2017YFA0700800) and Beijing Academy of Artificial Intelligence (BAAI).

NAS [28] are required for evaluation. For Eq. 2 step such corresponding labels are also necessary in the training set to optimize the weights. While in *convergence-based NAS*, Eq. 3 only depends on a metric to estimate the convergence speed, which is free of labels. Though the optimization in Eq. 4 still needs labels, the purpose of the training is just to provide the evidence for the benchmark in Eq. 3 rather than accurate representations. So, we conclude that in convergence-based NAS the requirement of labels is much weaker than that in performance-based NAS.

The observation motivates us to take a further step: in convergence-based NAS, *can we use only random labels for search, instead of any feasible labels like ground truths or pretext task labels entirely?* To demonstrate it, we propose a novel convergence-based NAS framework, called *Random Label NAS (RLNAS)*, which only requires random labels to search. RLNAS follows the paradigm of Eq. 3, 4. In Eq. 4 step, random labels are adopted to optimize the weight for each sampled architecture a; while in Eq. 3 step, a customized *angle* metric [21] is introduced to measure the distance between trained and initialized weights, which estimates the convergence speed of the corresponding architecture. To speed up the search procedure, RLNAS further utilizes the mechanism of *One-Shot NAS* [3, 19] to decouple the nested optimization of Eq. 3 and Eq. 4 into a *two-step* pipeline: first training a *SuperNet* with random labels, then extracting the sub-network with the fastest convergence speed from the SuperNet using evolutionary search.

We evaluate our RLNAS in popular search spaces like NAS-Bench-201 [15], DARTS [30] and MobileNet-like search space [5]. Very surprisingly, though RLNAS does not use any feasible labels, it still achieves comparable or even better performances on multiple benchmarks than many supervised/unsupervised methods, including state-of-the-art NAS frameworks such as *PC-DARTS* [43], *Single-Path One-Shot* [19], *FairDARTS* [13], *FBNet* [40] and *UnNAS* [28]. Moreover, networks discovered by RLNAS are also demonstrated to transfer well in the downstream tasks such as object detection and semantic segmentation.

In conclusion, the major contribution of the paper is that we propose a new convergence-based NAS framework *RLNAS*, which makes it possible to search with only random labels. We believe the potential of RLNAS may includes:

A simple but stronger baseline. Compared with the widely used *random search* [24] baseline, RLNAS is much more powerful, which can provide a stricter validation for future NAS algorithms.

Inspiring new understandings on NAS. Since the performance of RLNAS is as good as many supervised NAS frameworks, on one hand, it further validates the effectiveness of *ease-of-convergence hypothesis*. On the other hand, however, it suggests that the ground truth labels or NAS on specified tasks do not help much for current NAS algorithms, which implies that architectures found by existing NAS methods may still be suboptimal.

2. Related Work

Supervised Neural Architecture Search. Supervised neural architecture search (NAS) paradigm is the mainstream NAS setting. Looking back the development history, supervised NAS can be divided into two categories hierarchically: *nested NAS* and *weight-sharing NAS* from the perspective of search efficiency. In the early stage, *nested NAS* [49, 2, 50, 47, 48, 29, 35, 38] trains candidate architectures from scratch and update controller with corresponding performance feedbacks iteratively. However, *nested NAS* works at the cost of a surge in computation, e.g. NASNet [50] costs about 1350–1800 GPU days. ENAS [34] observes the computation bottleneck of *nested NAS* and forces all candidate architectures to share weights. ENAS takes $1000\times$ less computation cost than *nested NAS* [34] and proposes a new NAS paradigm named *weight-sharing NAS*. A large number of literature [30, 9, 43, 3, 4, 5, 19] follow the *weight-sharing* strategy due to the superiority of search efficiency. This work is also carried out under the *weight-sharing* strategy. Unlike most *weight-sharing* approaches, we are not focusing on the improvement of search efficiency.

According to different optimization steps, *weight-sharing* approaches can be further divided into two categories: the one joint step optimization approach named *gradient-based NAS* [30, 9, 43]) and the two sequential steps optimization approach named *One-Shot NAS* [3, 4, 5, 19]). The *gradient-based NAS* relaxes discrete search space into a continuous one with architecture parameters, which are optimized with end-to-end paradigms. Because of the non-differentiable characteristic of angle, we follow the mechanism of *One-Shot NAS* to study *convergence-based NAS*.

Unsupervised Neural Architecture Search. Recently, unsupervised learning [20, 8, 18] has received much attention, and the unsupervised paradigm has also appeared in the field of NAS. [44] used unsupervised architecture representation in the latent space to better distinguish network architectures with different performance. UnNAS [28] introduces unsupervised methods [17, 33, 46] to *weight-sharing NAS* in order to ablate the role of labels. Although UnNAS does not use the labels of the target dataset, the labels like *rotation category*, etc on the pretext tasks are still exploited. UnNAS shows that *weight-sharing NAS* can still work with the absence of ground truth labels, but it is hard to conclude that labels are completely unnecessary. Different from unsupervised learning, which requires representation, unsupervised NAS focuses on architectures. Therefore, random

labels are introduced in this paper, which completely detach from prior supervision information and help us thoroughly ablate the impact of labels on NAS.

Model Evaluation Metrics. [32, 1] develop *training-free NAS* which means searching directly at initialization without involving any training. They focus on investigating training-free model evaluation metrics to rank candidate architectures. [32] uses the correlation between input Jacobian to indicate model performance. [1] uses the combination of NTKs and linear regions in input space to measure the architecture trainability and expressivity. Although *training-free NAS* has much higher search efficiency, there is still a performance gap compared with well-trained *weight-sharing NAS*. ABS [21] introduces angle metric to indicate model performance and mainly focuses on search space shrinking. Different from ABS, we directly search architectures with angle metric.

3. Methodology

As mentioned in the introduction, in order to utilize the mechanism of Oner-Shot NAS, we first briefly review Single Path One-Shot (SPOS) [19] as preliminary. Based on SPOS framework, we then put forward our approach Random Label NAS (RLNAS).

3.1. Preliminary: SPOS

SPOS is one of the One-Shot approaches, which decouple the NAS optimization problem into two sequential steps: firstly train SuperNet, and then search architectures. Different from other One-Shot approaches, SPOS further decouples weights of candidate architectures by training SuperNet stochastically. Specifically, SPOS regards a candidate architecture in SuperNet as a single path and uniformly activates a single path to optimize corresponding weights in each iteration. Thus, the SuperNet training step can be expressed as:

$$\mathbf{W}_a^\star = \underset{\mathbf{W}}{\operatorname{argmin}}\, \mathbb{E}_{a\sim\Gamma(\mathcal{A})}\mathcal{L}\left(a, \mathbf{W}\right), \tag{5}$$

where $\mathcal{L}$ means objective function optimized on training dataset with ground truth labels and $\Gamma(\mathcal{A})$ is a uniform distribution of $a \in \mathcal{A}$.

After SuperNet trained to convergence, SPOS performs architecture search as:

$$a^\star = \underset{a\in\mathcal{A}}{\operatorname{argmax}}\, \mathrm{ACC}_{val}\left(a, \mathbf{W}_a^\star\right). \tag{6}$$

SPOS implements Eq. 6 by utilizing an evolution algorithm to search architectures. With initialized population, SPOS conducts crossover and mutation to generate new candidate architectures and uses validation accuracy as fitness to keep candidate architectures with top performance. Repeat this way until the evolution algorithm converges to the optimal architecture.

3.2. Our approach: Random Label NAS (RLNAS)

The combination of two decoupled optimization steps, SuperNet structure consisting of single paths and evolution search, makes SPOS simple but flexible. Following the mechanism of SPOS, we decouple the *convergence-based* optimization of Eq. 3 and Eq. 4 into the following two steps.

Firstly, SuperNet is trained with random labels:

$$\mathbf{W}_a^\star = \underset{\mathbf{W}}{\operatorname{argmin}}\, \mathbb{E}_{a\sim\Gamma(\mathcal{A})}\mathcal{L}\left(a, \mathbf{W}, R\right), \tag{7}$$

where R represents random labels; other notations follow the same definitions as in Eq. 5.

Secondly, evolution algorithm with convergence-based metric $\mathrm{Convergence}(\cdot)$ as fitness searches the optimal architecture from SuperNet:

$$a^\star = \underset{a\in\mathcal{A}}{\operatorname{argmax}}\, \mathrm{Convergence}\left(a, \mathbf{W}_a^\star\right). \tag{8}$$

In the next section, we introduce the mechanism of generating random labels in Sec. 3.2.1 and use an angle-based metric as $\mathrm{Convergence}(\cdot)$ to estimate model convergence speed in Sec. 3.2.2.

3.2.1 Random Labels Mechanism

In representation learning field, deep neural networks (DNNs) have the capacity to fit dataset with partial random labels [45]. Further more, [31] tries to understand what DNNs learn when trained on natural images with entirely random labels and experimentally demonstrates that pre-training on purely random labels can accelerate the training of downstream tasks under certain conditions. For NAS field, although we pursue the optimal model architecture rather than model representation in search phase, model representation is still involved in the performance-based NAS. However, it is still an open question can neural architecture search work within random labels setting. In the view of this, we try to study the impact of random labels on NAS optimization problem.

At first, we introduce the mechanism of generating random labels. To be specific, random labels obey the discrete uniform distribution and the number of discrete variable is equal to the image category of dataset in default (other possible methods are discussed in Sec. 4.3). Random labels corresponding to different images are sampled in data preprocessing procedure and these image-label pairs will not change during the whole model optimization process.

3.2.2 Angle-based Model Evaluation Metric

Recently, [37] found out that searched architectures by NAS algorithms share the same pattern of fast convergence. With

this rule as a breach, we try to design model evaluation metrics from the perspective of model convergence. [6] firstly measure the convergence of a stand-alone trained model with a angle-based metric. The metric is defined as the angle between initial model wights and trained ones. ABS [21] introduces this metric into the NAS community and uses it to shrink the search space progressively. Different from ABS, we focus on the optimization problem with random labels and adopt angle-based metric to directly search architectures rather than shrink search space. Prior to extend angle to guide architecture search, we first review angle metric in ABS [21].

Review Angle Metric in ABS. SuperNet is represented as a directed acyclic graph (DAG) denoted as $\mathcal{A}(\mathbf{O}, \mathbf{E})$, where $\mathbf{O}$ is the set of feature nodes and $\mathbf{E}$ is the set of connections (each connection is instantiated as an alternative operation) between two feature nodes. ABS defines $\mathcal{A}(\mathbf{O}, \mathbf{E})$ with the only input node O_{in} and the only output node O_{out}. A candidate architecture is sampled from SuperNet and it is represented as $a(\mathbf{O}, \widetilde{\mathbf{E}})$. The candidate architecture has the same feature nodes $\mathbf{O}$ as SuperNet but subset edges $\widetilde{\mathbf{E}} \in \mathbf{E}$. ABS uses a weight vector $\boldsymbol{V}(a, \mathbf{W})$ to represent a model and constructs $\boldsymbol{V}(a, \mathbf{W})$ by concatenating the weights of all paths from O_{in} to O_{out}. The distance between the initialized candidate architecture whose weights is $\mathbf{W_0}$ and the trained one with weights $\mathbf{W_t}$ is:

$$\text{Angle}(a) = \arccos\left(\frac{< \boldsymbol{V}(a, \mathbf{W}_0), \boldsymbol{V}(a, \mathbf{W}_t) >}{\|\boldsymbol{V}(a, \mathbf{W}_0)\|_2 \cdot \|\boldsymbol{V}(a, \mathbf{W}_t)\|_2}\right). \quad (9)$$

Extensive Representation of Weight Vector. As above discussed, ABS define the SuperNet with just one input node and one output node. However, for some search spaces, they consist of cell structures with multiple input nodes and outputs nodes. For example, each cell in DARTS has two input nodes and the output node of each cell consists outputs of all intermediate nodes by concatenation, which motivates us to consider all intermediate nodes as output nodes for the identification of architecture topology. In general, we redefine weight vector $\boldsymbol{V}(a, \mathbf{W})$ by concatenating the weights of all paths from $\mathbf{O}_{in}$ to $\mathbf{O}_{out}$.

Parameterize Non-weight Operations. So as to resolve the conflict among candidate architectures with the same learnable weights, ABS parameterizes non-weight operations ('pool', 'skip-connect' and 'none'). The 'pool' operation (both 'average pool' and 'max pool') is assigned with a fixed tensor with dimension $[O, C, K, K]$ (O and C represent output channels and input channels respectively, K is the kernel size) and all elements are $1/K^2$. Different from ABS assign 'skip-connect' with empty vector, we propose an alternative parametric method, which assigns identity tensor with dimension $[O, C, 1, 1]$ to the 'skip-connect' operation. We adjust parametric methods for different search spaces, e.g., empty weights and identity tensor are assigned to 'skip-connect' in NAS-Bench-201 and DARTS or MobileNet-like search space respectively. The reason for the difference may be related to the complexity of the search space. The 'none' operation need not to be parameterized as ABS and it determines the number of paths that make up the weights vector $\boldsymbol{V}$. If there is a 'none' in a path, then weights of operations in this path will not involved in angle calculation.

4. Experiments

4.1. Search Space and Training Setting

We analyze and evaluate RLNAS on three existing popular search spaces: NAS-Bench-201 [15], DARTS [30] and MobileNet-like search space [5].

NAS-Bench-201. There are 6 edges in each cell and each edge has 5 alternative operations. Because of repeated stacking, NAS-Bench-201 consists of 15625 candidate architectures and provides the real performance for each architecture. We adopt the same training setting for SuperNet in a single GPU across CIFAR-10 [23] CIFAR-100 [23] and ImageNet16-120 [11]. We train the SuperNet 250 epochs with mini-batch 64. We use SGD to optimize weights with momentum 0.9 and weight decay $5e^{-4}$. The learning rate follows cosine schedule from initial 0.025 annealed to 0.001. In evolution phase, we use population size 100, max iterations 20 and keep top-30 architectures in each iteration. All experiment results on NAS-Bench-201 are obtained in three independent runs with different random seeds.

DARTS. Different from vanilla DARTS [30], each intermediate node only samples two operations among alternative operations (except 'none') from its all preceding nodes in SuperNet training phase. We train the SuperNet with 8 cell on CIFAR-10 for 250 epochs and other training settings keep the same as DARTS [30]. We also train 14 cell SuperNet with initial channel 48 on ImageNet. We use 8 GPUs to train SuperNet 50 epochs with mini-batch 512. SGD with momentum 0.9 and weight decay $4e^{-5}$ is adopted to optimize weights. The cosine learning rate schedules from 0.1 to $5e^{-4}$. We use the same evolution hyper-parameters as Single Path One-Shot (SPOS) [19]. As for model evaluation phase (retrain searched architecture), we follow the training setting as PC-DARTS [43] on ImageNet.

MobileNet. The MobileNet-like search space proposed in ProxylessNAS [5] is adopted in this paper. The SuperNet contains 21 choice blocks and each block has 7 alternatives:

Method	Configurations		CIFAR-10 (%)		CIFAR-100 (%)		ImageNet16-120 (%)	
	Label type	Performance indicator	valid acc	test acc	valid acc	test acc	valid acc	test acc
A (SPOS)	ground truth label	validation accuracy	88.49	92.11	66.51	66.89	40.16	40.80
B	ground truth label	angle	**90.20**	**93.76**	70.71	**71.11**	40.78	41.44
C	random label	validation accuracy	76.47	80.60	52.48	52.84	29.58	28.37
D (RLNAS)	random label	angle	89.94	93.45	**70.98**	70.71	**43.86**	**43.70**

Table 1: Search performance on NAS-Bench-201 across CIFAR-10, CIFAR-100 and ImageNet16-120.

6 MobileNet blocks (combination of kernel size {3,5,7} and expand ratio {3,6}) and 'skip-connect'. We keep the same experiment setting for both search phase and evaluation phase as SPOS [19].

4.2. Searching Results

4.2.1 NAS-Bench-201 Experiment Results

Search performance. For NAS-Bench-201 search space, experiments are conducted on three datasets: CIFAR-10, CIFAR-100 and ImageNet16-120. Different from other literature only search on CIFAR-10 and look up real performance of the found architecture on various test dataset (e.g., test accuracy on CIFAR-100 or ImageNet16-120), we actually train SuperNet on different target datasets and search architectures with the unique SuperNet. Firstly, we construct SuperNet based NAS-Bench-201 search space and train the SuperNet by uniform sampling strategy [19] with ground truth labels or random labels. Then, angle or validation accuracy is regarded as fitness to perform evolution search. According to different method configurations, there are total four possible methods as described in Table 1. For simplity, we denoted they as method A, B, C and D respectively. In particular, method A and D correspond to SPOS and RLNAS. The search performance on three datasets are reported in Table 1. We first compare method C and D within the random label setting, and find that angle surpasses validation accuracy with a large margin. Similar results can also be observed under the ground truth label setting, but the margin between method A and B is not such large. This suggests that angle can evaluate models more accurately than validation accuracy. Further more, in the case where angle is used as the metric, even if random labels are used, RLNAS obtains comparable accuracy on CIFAR-10 and CIFAR-100 and even outperforms method B by 1.26% test accuracy on ImageNet16-120.

Ranking correlation. In addition to the analysis of top architectures as Table 1, we further conduct rank correlation analysis. The first step is also to train SuperNet with ground truth labels or random labels. Secondly, we traverse the whole NAS-Bench-201 search space and rank them with different model evaluation metrics independently. We treat the rank based on real performance provided by NAS-Bench-201 as the ground truth rank. At last, we compute the Kendall's Tau [22, 36, 12, 21] between the rank based on the model evaluation metric and the ground truth rank to evaluate the ranking correlation. We compare angle and validation accuracy as model evaluation metric in both ground truth label and random label setting across three datasets. The ranking correlation results are shown in Table 2. The results on different datasets show the consistent order of ranking correlation: C<A<D<B. It should be noted that the rank obtained by validation accuracy in the case of random labels has almost no correlation with the ground truth rank. To our surprise, angle still has the ranking correlation around 0.5 under the random label setting, which even exceeds validation accuracy in ground truth label case.

Method†	CIFAR-10	CIFAR-100	ImageNet16-120
A (SPOS)	0.4239	0.4832	0.4322
B	**0.6671**	**0.6942**	**0.6342**
C	0.0874	−0.0195	−0.0262
D (RLNAS)	0.5059	0.5097	0.4716

Table 2: Ranking correlation on NAS-Bench-201. † refer to Table 1 for detailed method configurations.

4.2.2 DARTS Search Space Results

We conduct two types of experiments in DARTS search space: search architectures with 8 cells on CIFAR-10, then transfer to ImageNet and search architectures with 14 cells on ImageNet directly. For experiment conducted on CIFAR-10, the training dataset is divided into two subsets with equal size, one of which is used to train the SuperNet, and the other is used as the validation dataset to evaluate model performance in the search phase. As for experiments searched on ImageNet, 50K images are separated from the original training dataset as validation and the rest images are used as the new training dataset.

Search architectures on CIFAR-10. We first analyze the search performance on CIFAR-10 dataset in Table 3. RLNAS embodies strong generalization ability when transfering searched architecture from CIFAR-10 to ImageNet. As shown in the first block of Table 3, RLNAS has reached 76.0% top-1 accuracy, even obtains 75.6% within 600M FLOPs constrain.

Search type	Method	Params (M)	FLOPs (M)	Top-1 (%)	Top-5 (%)
CIFAR-10	DARTS [30] (*sup.*)	4.7	574	73.3	91.3
	SPOS [19] (*sup, our impl.*)	4.3	471	73.7	91.6
	PC-DARTS [43] (*sup.*)	5.3	586	74.9	92.2
	FairDARTS-B [13] (*sup.*)	4.8	541	75.1	92.5
	P-DARTS [9] (*sup.*)	4.9	557	75.6	92.6
	RLNAS (*unsup.*)	5.7	629	**76.0**	**92.9**
	RLNAS$^{\blacktriangledown}$ (*unsup.*)	5.3	581	75.6	92.5
ImageNet	SPOS [19] (*sup, our impl.*)	4.6	512	74.5	92.1
	NAS-DARTS† [28] (*sup.*)	5.3	582	**76.0**	92.7
	PC-DARTS [43] (*sup.*)	5.3	597	75.8	92.7
	RLNAS (*unsup.*)	5.5	597	75.9	**92.9**

Table 3: DARTS search space results: comparison of the SOTA methods on ImageNet. There are two search types of methods and the results of the first block and the second block are searched on CIFAR-10 and ImageNet respectively. $^{\blacktriangledown}$ FLOPs of the searched architecture is scaled down within 600M by adjusting initial channels from 48 to 46. † retrain NAS-DARTS reported in UnNAS [28] as PC-DARTS [43].

Search architectures on ImageNet. After demonstrating the transferring ability of RLNAS among classification tasks, we further verify the efficacy of our method by directly searching on ImageNet. To our best knowledge, it is the first time to train SuperNet with **14 cells** in DARTS search space without any SuperNet structure modification or complicated techniques. After SuperNet training, we search candidate architectures with 600M FLOPs constrain. The searching results are shown in the second block of Table 3 and RLNAS obtains 75.9%. Compared with the results found on CIFAR-10, the performance of RLNAS is further improved by 0.3%, which indicates that narrowing the gap between the training setting (both dataset and SuperNet structure) of the search phase and the one in the evaluation phase is helpful for architecture search.

Comparison with UnNAS. Further, we compare our method with UnNAS [28] which also search architectures directly on ImageNet-1K with three pretext tasks [17, 33, 46]. For fair comparisons with UnNAS, we have no FLOPs limit in the search phase, but after the search is completed, we limit the FLOPS within 600M by scaling the initial channels from 48 to 42. Simultaneously, we retrain the three architectures reported as UnNAS [28] with the same training setting as PC-DARTS [43]. Table 4 shows that our method obtains high performance with 76.7% and 75.9% within 600M FLOPs constrain, which is comparable with UnNAS with jigsaw task and competitive to results obtained by the other two pretext tasks.

4.2.3 MobileNet-like Search Space Results.

To verify the versatility of our method, we further conduct experiments in the MobileNet-like search space. We train SuperNet with 120 epochs on ImageNet as [19]. In the search phase, we limit model FLOPs within 475M so as to make fair comparisons with other methods. Results are summarized in Table 5. RLNAS obtains 75.6% top-1 accuracy. Compared with other SOTA methods, our method even outperforms with a slight margin, which verify that our strategy does not overfit to any search space and can achieve effective results generally.

Method	Params (M)	FLOPs (M)	Top-1 (%)	Top-5 (%)
UnNAS [28] (*rotation task.*)	5.1	552	75.8	92.6
UnNAS [28] (*color task.*)	5.3	587	75.5	92.6
UnNAS [28] (*jigsaw task.*)	5.2	560	76.2	92.8
RLNAS (*random label.*)	6.6	724	**76.7**	**93.1**
RLNAS$^{\blacktriangledown}$ (*random label.*)	5.2	561	75.9	92.8

Table 4: DARTS search space results: comparison with UnNAS on ImageNet. The architectures of UnNAS based on three pretext tasks are provided in [28] and we retrain them as PC-DARTS training setting [43].$^{\blacktriangledown}$ FLOPs of the searched architecture is scaled down within 600M by adjusting initial channels from 48 to 42.

Method	Params (M)	FLOPs (M)	Top-1 (%)	Top-5 (%)
FairNAS-A [12] (*sup.*)	4.6	388	75.3	92.4
FBNet-C [40] (*sup.*)	4.4	375	74.9	92.1
Proxyless (GPU) [5] (*sup.*)	7.0	457	75.1	92.5
FairDARTS-D [13] (*sup.*)	4.3	440	**75.6**	**92.6**
SPOS [19] (*sup.*)	5.4	472	74.8	-
RLNAS (*unsup.*)	5.3	473	**75.6**	**92.6**

Table 5: MobileNet search space results: comparison of the SOTA methods on ImageNet.

4.3. Ablation Study and Analysis

We perform ablation study in this section. We analyze the impact of random labels and angle metric on RLNAS. All experiments are conducted on NAS-Bench-201.

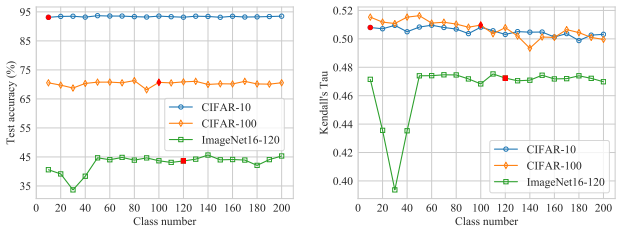
Methods of generating random labels. In the above experiments, we uniformly sample random labels for images before SuperNet training and we denote it as (1). In this subsection, we further discuss 3 other methods for generating random labels: (2). shuffle all ground truth labels at once before SuperNet training, (3). uniformly sample labels in each training iteration, and (4). shuffle ground truth labels in each training iteration. According to these four methods, we conducted three repeated architecture search experiments across CIFAR-10, CIFAR-100 and ImageNet16-120.

As Table 6 shows, in general, the methods of generating random labels at one time have higher performance than the methods of randomly generating labels in each iteration. Even if RLNAS† has better performance than RLNAS* and RLNAS* on CIFAR-10 and CIFAR-100, the performance on ImageNet16-120 is poor with a large margin and this means that RLNAS† is instable and has poor transferring ability. As for RLNAS* and RLNAS*, these two methods obtain comparable test accuracy. Considering RLNAS* coupled with ground truth labels, we generate random labels with RLNAS* in default and it is easy to apply our algorithm to tasks without labels.

Method	CIFAR-10 test acc (%)	CIFAR-100 test acc (%)	ImageNet16-120 test acc (%)
RLNAS*	93.45±0.11	70.71±0.36	43.70±1.25
RLNAS*	93.52±0.27	70.25±0.25	**43.81±1.12**
RLNAS†	92.85±0.46	61.59±6.57	27.51±1.04
RLNAS‡	**93.65±0.07**	**71.45±0.42**	27.51±1.04

Table 6: Search results of four generating random label method on NAS-Bench-201: (1).* uniform sample all random labels at once, (2).* shuffle all ground truth labels at once, (3).† uniform sample labels in each iteration, and (4).‡shuffle ground truth labels in each iteration.

Impact of image category. We have shown that uniform sample labels corresponding images before training is the most appropriate method to generate random labels. In this section, we further discuss the impact of the label category on searching performance. In detail, we sample 20 different categories from 10 to 200 with interval 10 for CIFAR-10, CIFAR-100 and ImageNet16-120. SuperNet is trained with different categories of random labels. After that, test accuracy and Kendall's Tau are obtained like subsection 4.2.1. As shown in Figure 1, test accuracy and Kendall's Tau fluctuate greatly when the number of categories on the ImageNet16-120 is small (in $[10, 50]$). However, Kendall's Tau and test accuracy are not sensitive to label categories in most cases. This observation implies that our method can be directly applied to tasks where the real image category is unknown.



(a) Test accuracy (b) Kendall's Tau

Figure 1: Impact of the random label category on (a) test accuracy and (b) Kendall's Tau (best view in color). CIFAR-10, CIFAR-100 and ImageNet16-120 all sample 20 different image categories from 10 to 200 with interval 10. The red marker in each polyline represents the number of real image categories for different datasets.

Bias analysis of angle metric. We have shown the impacts of random labels on RLNAS in the above section. Next, we further ablate the bias of angle metric in architecture search. Specifically, we initialize two SuperNet weights with the same distribution but different random seeds. Based on the SuperNet without training, evolution algorithm with angle is used to search architectures. We also construct a random search baseline which train SuperNet with uniform sampling strategy and ground truth labels, then randomly sample 100 architectures from NAS-Bench-201 search space. The top-1 architecture is selected among the sampled architectures according to their validation accuracy. Table 7 compares our method with two training free methods with different initialization and one random search method. The results show that the two training free methods are worse than random search, and RLNAS is better than random search. This means that angle metric will not bias to a certain candidate architecture.

Method	CIFAR-10 test acc (%)	CIFAR-100 test acc (%)	ImageNet16-120 test acc (%)
Training free†	90.74±1.39	66.97±1.86	38.54±2.86
Training free‡	91.55±1.34	66.59±2.10	39.03±3.91
Random search	92.09±0.21	67.27±1.28	40.77±3.64
RLNAS	**93.45±0.11**	**70.71±0.36**	**43.70±1.25**

Table 7: Bias analysis of angle towards architectures on NAS-Bench-201.† and ‡ initializes model weights with normalization distribution and uniform distribution.

4.4. Generalization Ability

We evaluate the generalization ability of RLNAS on two downstream tasks: object detection and semantic segmentation. We first retrain the models searched by different NAS methods on ImageNet , and then finetune these pre-trained

Method	Params (M)	FLOPs (M)	Acc	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Random search	4.7	519	74.3	31.7	50.4	33.4	16.3	35.2	42.9
DARTS-v1 [30] (*sup.*)	4.5	507	74.3	31.2	49.5	32.6	16.1	33.9	43.6
DARTS-v2 [30] (*sup.*)	4.7	531	74.9	31.5	50.3	33.1	16.9	34.5	43.0
P-DARTS [9] (*sup.*)	4.9	544	75.7	32.9	**52.1**	34.7	17.2	36.2	44.8
PC-DARTS [43] (*sup.*)	5.3	582	75.9	32.9	51.8	**34.8**	**17.5**	36.3	43.5
UnNAS [28] (*rotation task.*)	5.1	552	75.8	32.8	51.5	34.7	16.7	36.1	44.5
UnNAS [28] (*color task.*)	5.3	587	75.5	32.4	51.2	34.2	16.6	35.6	44.6
UnNAS [28] (*jigsaw task.*)	5.2	560	76.2	**33.0**	51.9	35.3	16.4	**37.2**	**45.4**
Ours† (*random label.*)	5.5	597	75.9	32.4	50.9	34.4	16.5	35.5	44.5
Ours‡ (*random label.*)	5.2	561	75.9	32.9	51.6	**34.8**	16.8	36.7	44.5

Table 8: Object detection results of DARTS search space on MS COCO. † search with 600M FLOPs constrain. ‡ search without FLOPs constrain but scale FLOPs to 600M.

Method	Params (M)	FLOPs (M)	Acc	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Random search (*sup.*)	4.5	446	75.3	29.7	47.5	31.4	15.3	32.6	39.9
FairNAS-A [12] (*sup.*)	4.7	389	75.1	29.8	47.8	31.4	15.5	32.3	**41.2**
Proxyless (GPU) [5] (*sup.*)	7.0	457	75.5	29.5	47.5	30.9	15.5	32.4	40.8
FairDARTS-D [13] (*sup.*)	4.4	477	74.7	29.6	47.2	31.1	14.6	32.5	40.1
SPOS [19] (*sup.*)	5.4	472	75.6	29.8	**48.1**	31.1	**16.0**	32.6	40.4
Ours (*unsup.*)	5.3	473	75.6	**30.0**	47.6	**31.8**	15.7	**32.8**	40.5

Table 9: Object detection results of MobileNet-like search space on MS COCO.

models on downstream tasks. In order to make fair comparisons, models searched in the same search space adopt the same training setting for ImageNet classification tasks. At the same time, models for the same downstream task also use the same training setting, no matter what search space the model is searched from.

Object detection. We conduct experiments on MS COCO [26] and adopt RetinaNet [25] as the detection framework. The train and test image scale is 800× resolution. We only modify the backbone of RetinaNet and train RetinaNet with default training setting as Detectron2 [41]. Table 8 and Table 9 show the comparisons of models searched in DARTS and MobileNet-like search space respectively. RLNAS obtains comparable AP in DARTS search space and surpasses other methods with slight margin in MobileNet-like search space.

Semantic segmentation. We further test RLNAS on the task of semantic segmentation on Cityscapes [14] dataset. We adopt DeepLab-v3 [7] as segmentation framework. The train and test image scale is 769×769 and we train DeepLab-v3 with 40k iterations. The other segmentation training setting are kept the same as MMSegmentation [42]. Table 10 and Table 11 make comparisons among models searched on DARTS and MobileNet-like search space respectively. For DARTS search space, RLNAS† obtains 73.2% mIoU and outperform other methods by a large margin. RLNAS also obtains comparable mIoU compared to other methods in MobileNet search space.

Summary. We conclude that RLNAS achieves comparable or even superior performance across two downstream tasks and various search spaces, without bells and whistles.

Method	Params (M)	FLOPs (M)	Acc (%)	mIoU (%)
Random search (*sup.*)	4.7	519	74.3	72.3
DARTS-v1 [30] (*sup.*)	4.5	507	74.3	72.7
DARTS-v2 [30] (*sup.*)	4.7	531	74.9	71.8
P-DARTS [43] (*sup.*)	4.9	544	75.7	71.9
PC-DARTS [43] (*sup.*)	5.3	582	75.9	72.2
UnNAS [28] (*rotation task.*)	5.1	552	75.8	71.9
UnNAS [28] (*color task.*)	5.3	587	75.5	72.0
UnNAS [28] (*jigsaw task.*)	5.2	560	76.2	72.1
Ours† (*random label.*)	5.5	597	75.9	**73.2**
Ours‡ (*random label.*)	5.2	561	75.9	72.5

Table 10: Semantic segmentation results of DARTS search space on Cityscapes.† search with 600M FLOPs constrain. ‡ search without FLOPs constrain but scale FLOPs to 600M.

Method	Params (M)	FLOPs (M)	Acc (%)	mIoU (%)
Random search (*sup.*)	4.5	446	75.3	70.6
FairNAS-A [12] (*sup.*)	4.7	389	75.1	72.0
Proxyless (GPU) [5] (*sup.*)	7.0	457	75.5	71.0
FairDARTS-D [13] (*sup.*)	4.4	477	74.7	**72.1**
SPOS [19] (*sup.*)	5.4	472	75.6	71.6
Ours (*unsup.*)	5.3	473	75.6	71.8

Table 11: Semantic segmentation results of MobileNet-like search space on Cityscapes.

References

[1] Anonymous. Neural architecture search on imagenet in four {gpu} hours: A theoretically inspired perspective. In *Submitted to International Conference on Learning Representations*, 2021. under review. 3

[2] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing neural network architectures using reinforcement learning. *arXiv preprint arXiv:1611.02167*, 2016. 1, 2

[3] Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc Le. Understanding and simplifying one-shot architecture search. In *International Conference on Machine Learning*, pages 550–559, 2018. 2

[4] Andrew Brock, Theodore Lim, James M Ritchie, and Nick Weston. Smash: one-shot model architecture search through hypernetworks. *arXiv preprint arXiv:1708.05344*, 2017. 2

[5] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. *arXiv preprint arXiv:1812.00332*, 2018. 2, 4, 6, 8

[6] Simon Carbonnelle and Christophe De Vleeschouwer. Layer rotation: a surprisingly powerful indicator of generalization in deep networks? *arXiv preprint arXiv:1806.01603*, 2018. 4

[7] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017. 8

[8] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020. 2

[9] Xin Chen, Lingxi Xie, Jun Wu, and Qi Tian. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1294–1303, 2019. 2, 6, 8

[10] Yukang Chen, Tong Yang, Xiangyu Zhang, Gaofeng Meng, Xinyu Xiao, and Jian Sun. Detnas: Backbone search for object detection. In *Advances in Neural Information Processing Systems*, pages 6638–6648, 2019. 1

[11] Patryk Chrabaszcz, Ilya Loshchilov, and Frank Hutter. A downsampled variant of imagenet as an alternative to the cifar datasets. *arXiv preprint arXiv:1707.08819*, 2017. 4

[12] Xiangxiang Chu, Bo Zhang, Ruijun Xu, and Jixiang Li. Fairnas: Rethinking evaluation fairness of weight sharing neural architecture search. *arXiv preprint arXiv:1907.01845*, 2019. 5, 6, 8

[13] Xiangxiang Chu, Tianbao Zhou, Bo Zhang, and Jixiang Li. Fair darts: Eliminating unfair advantages in differentiable architecture search. *arXiv preprint arXiv:1911.12126*, 2019. 2, 6, 8

[14] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. *CVPR*, 2016. 8

[15] Xuanyi Dong and Yi Yang. Nas-bench-102: Extending the scope of reproducible neural architecture search. *arXiv preprint arXiv:2001.00326*, 2020. 2, 4

[16] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Nas-fpn: Learning scalable feature pyramid architecture for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7036–7045, 2019. 1

[17] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. *arXiv preprint arXiv:1803.07728*, 2018. 1, 2, 6

[18] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised learning, 2020. 2

[19] Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. *arXiv preprint arXiv:1904.00420*, 2019. 1, 2, 3, 4, 5, 6, 8

[20] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 2

[21] Yiming Hu, Yuding Liang, Zichao Guo, Ruosi Wan, Xiangyu Zhang, Yichen Wei, Qingyi Gu, and Jian Sun. Angle-based search space shrinking for neural architecture search. *arXiv preprint arXiv:2004.13431*, 2020. 2, 3, 4, 5

[22] Maurice G Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938. 5

[23] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *Tech Report*, 2009. 4

[24] Liam Li and Ameet Talwalkar. Random search and reproducibility for neural architecture search. In *Uncertainty in Artificial Intelligence*, pages 367–377. PMLR, 2020. 2

[25] Tsung Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2999–3007, 2017. 8

[26] Tsung Yi Lin, Michael Maire, Serge Belongie, James Hays, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, 2014. 8

[27] Chenxi Liu, Liang-Chieh Chen, Florian Schroff, Hartwig Adam, Wei Hua, Alan L Yuille, and Li Fei-Fei. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 82–92, 2019. 1

[28] Chenxi Liu, Piotr Dollár, Kaiming He, Ross Girshick, Alan Yuille, and Saining Xie. Are labels necessary for neural architecture search? *arXiv preprint arXiv:2003.12056*, 2020. 1, 2, 6, 8

[29] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture

search. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 19–34, 2018. 1, 2

[30] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018. 2, 4, 6, 8

[31] Hartmut Maennel, Ibrahim Alabdulmohsin, Ilya Tolstikhin, Robert JN Baldock, Olivier Bousquet, Sylvain Gelly, and Daniel Keysers. What do neural networks learn when trained with random labels? *arXiv preprint arXiv:2006.10455*, 2020. 3

[32] Joseph Mellor, Jack Turner, Amos Storkey, and Elliot J Crowley. Neural architecture search without training. *arXiv preprint arXiv:2006.04647*, 2020. 3

[33] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European Conference on Computer Vision*, pages 69–84. Springer, 2016. 2, 6

[34] Hieu Pham, Melody Y Guan, Barret Zoph, Quoc V Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. *arXiv preprint arXiv:1802.03268*, 2018. 2

[35] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pages 4780–4789, 2019. 1, 2

[36] Christian Sciuto, Kaicheng Yu, Martin Jaggi, Claudiu Musat, and Mathieu Salzmann. Evaluating the search phase of neural architecture search. *arXiv preprint arXiv:1902.08142*, 2(3), 2019. 5

[37] Yao Shu, Wei Wang, and Shaofeng Cai. Understanding architectures learnt by cell-based neural architecture search. In *International Conference on Learning Representations*, 2020. 1, 3

[38] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2820–2828, 2019. 1, 2

[39] Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019. 1

[40] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10734–10742, 2019. 2, 6

[41] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019. 8

[42] Jiarui Xu, Kai Chen, and Dahua Lin. MMSegmenation. https://github.com/open-mmlab/mmsegmentation, 2020. 8

[43] Yuhui Xu, Lingxi Xie, Xiaopeng Zhang, Xin Chen, Guo-Jun Qi, Qi Tian, and Hongkai Xiong. Pc-darts: Partial channel connections for memory-efficient differentiable architecture search. *arXiv preprint arXiv:1907.05737*, 2019. 2, 4, 6, 8

[44] Shen Yan, Yu Zheng, Wei Ao, Xiao Zeng, and Mi Zhang. Does unsupervised architecture representation learning help neural architecture search? *arXiv preprint arXiv:2006.06936*, 2020. 2

[45] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016. 3

[46] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *European conference on computer vision*, pages 649–666. Springer, 2016. 2, 6

[47] Zhao Zhong, Junjie Yan, Wei Wu, Jing Shao, and Cheng-Lin Liu. Practical block-wise neural network architecture generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2423–2432, 2018. 1, 2

[48] Zhao Zhong, Zichen Yang, Boyang Deng, Junjie Yan, Wei Wu, Jing Shao, and Cheng-Lin Liu. Blockqnn: Efficient block-wise neural network architecture generation. *arXiv preprint arXiv:1808.05584*, 2018. 1, 2

[49] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016. 1, 2

[50] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710, 2018. 1, 2